
Single-Encoder VAE Image-To-Image Translation

Adit Gajjar

Harsh Nasit

Filip Balucha

Abstract

Image-to-image translation is a computer vision problem where an image is mapped from one domain to another. In the supervised scenario, the dataset consists of perfectly-paired image data. However, such data is either hard to find or simply does not exist [?]. Therefore, we focus on the unsupervised setting, where a joint distribution of images across two domains is learnt using the marginal distributions – the images in individual domains [?]. We propose a model that builds on the shared latent space assumption and uses domain-specific decoders. In addition, our approach readily extends to more than two different domains. The implementation is available at <https://github.com/HareshNasit/CSC413-Project>.

1 Introduction

At a high level, image-to-image translation aims to capture the characteristics of one image collection and determine how these could be translated into another image collection [?]. Mathematically, the goal is to learn a mapping $G : X \rightarrow Y$, where X is the source and Y the target domain.

Image-to-image translation includes many computer vision problems, such as super-resolution, colorization and style transfer [?]. Style transfer has various applications in photo and video editing, customized gaming styles and commercial art. Transferring styles between images is a hard image processing task because it is difficult to represent semantic information such as image texture.

Many efforts were put into the supervised setting, where models learn image-to-image translation across two domains using perfectly-paired images [?]. However, obtaining paired data is often difficult and expensive, and sometimes impossible [?]. For example, to learn the translation from photographs to images styled as Monet paintings, the supervised setting would require pairs of real-life images and the corresponding Monet paintings – an infeasible task. Therefore, we focus on unsupervised algorithms that map between domains without paired examples.

From a probabilistic perspective, the aim is to learn a joint distribution of images between two different domains, where each domain consists of images from a marginal distribution [?]. However, there are infinitely many such joint distributions [?]. To address this problem, we can add more structure to the objective [?] or make assumptions about the structure of the joint distribution [?].

For example, the CycleGAN model resolves this by training two mappings $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and updating the objective to enforce $F(G(x)) \approx x$ and $G(F(y)) \approx y$ [?]. By contrast, [?] assume a shared latent space, where a pair of corresponding images from different domains is mapped to the same latent representation.

Inspired by the stable training and meaningful encodings characteristic of variational autoencoder (VAE) models, we continue in the latter vein. We propose a model that leverages the shared latent space assumption but is simpler than the model proposed by [?]. While our model does not outperform theirs, we propose future directions that could bring our architecture to fruition.

2 Related works

Image-to-image translation is a well-studied problem. We provide a summary of papers that tackle the unsupervised setting. We focus on two well-known architectures – the popular Generative Adversarial Networks (GANs) and the Variational Autoencoders (VAEs), praised for learning useful representations that lend themselves to manipulation.

GANs. CycleGAN builds on the successful GAN architecture by introducing cycle-consistency. This constrains the image-to-image translation problem and encourages meaningful translations [?].

VAEs. Several methods use VAEs in the context of image-to-image translation or style transfer. ST-VAE performs multiple style transfer by projecting styles to a linear latent space and merging them using linear interpolation [?]. [?] propose a framework that explicitly models two representations – one for image content and another for style. [?] propose VAE-GAN, which combines a VAE and the discriminator from the GAN architecture. Finally, [?] apply the cycle-consistency idea to the VAE setting.

VAEs and Shared latent representation. Several research efforts establish a shared latent space. [?] propose a mapping between the two latent spaces corresponding to the two domains. Perhaps closest to our approach, [?] share the latent space z between the two domains by sharing some weights of domain-specific encoders and decoders. Our simplified architecture achieves the same using a single, shared encoder.

3 Method

3.1 Objective

The objective adapts the vanilla VAE objective that consists of a variational and a reconstruction term.

3.1.1 Variational loss

The variational loss term regularizes the latent space. It captures the KL divergence between the approximated posterior and the prior distribution. The posterior represents the “encoder” distribution, and the prior represents the family of distribution the encoder distribution should belong to. Similar to the original VAE paper, we assume both of these distributions are Gaussian [?]. This allows us to express the variational loss term as

$$\mathcal{L}_{\text{Variational}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \sum_{j=1}^J \left(1 + \log \left(\left(\sigma_j^{(i)} \right)^2 \right) - \left(\mu_j^{(i)} \right)^2 - \left(\sigma_j^{(i)} \right)^2 \right) \quad (1)$$

where σ and μ determine the approximated “encoder” distribution, J is the latent dimension, and i ranges over all samples in the batch \mathcal{B} .

3.1.2 Reconstruction loss

The original VAE paper uses a MSE reconstruction loss function [?]. However, in the context of images, this loss function assumes that the impact of noise is independent of the local characteristics [?]. We therefore use a multiscale structural similarity index (MS-SSIM) term. MS-SSIM is a differentiable similarity metric that takes into account local features at various scale [?] and aims to produce visually-pleasing outputs. However, we found that using MS-SSIM term alone led to slow convergence. Therefore, similar to [?], we instead augment the original MSE loss with MS-SSIM. We therefore express the reconstruction loss as

$$\mathcal{L}_{\text{Reconstruction}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \text{MSE} \left(\mathbf{u}^{(i)}, \tilde{\mathbf{u}}^{(i)} \right) + \text{MS-SSIM} \left(\mathbf{u}^{(i)}, \tilde{\mathbf{u}}^{(i)} \right) \quad (2)$$

3.1.3 Full loss

Finally, our full objective can be expressed as

$$\mathcal{L} = \mathcal{L}_{\text{Variational}} + \alpha \cdot \mathcal{L}_{\text{Reconstruction}} \quad (3)$$

where α is used to account for different learning rates between the two loss terms.

3.2 Architecture

Training 1: Encoder. Our architecture combines the components of the standard convolutional VAE (Figure ??). Both domains share a single encoder E , which is trained in concord with a shared decoder D . Its purpose is to establish a shared latent space where, based on the shared latent space assumption, images with similar content from different domains will be mapped to a similar point.




Figure 1: Learning a shared latent representation for both image domains.

Algorithm 1 The pre-training procedure

```
//  $X, Y \leftarrow \text{loadDataset}()$   
 $M \leftarrow \text{new VAE}()$   
 $E \leftarrow \text{new Encoder}()$   
 $M.\text{encoder} \leftarrow E$   
 $D \leftarrow \text{new Decoder}()$   
 $M.\text{decoder} \leftarrow D$   
 $M.\text{train}(X \cup Y)$ 
```

Training 2: Domain-specific decoders. Once the encoder is trained, the encoder weights are frozen to fix the mapping to a shared latent space. Then, a separate decoder is trained for each domain (Figure ??). For example, if we wish to map between images of apples and oranges, E would encode images of both, apples and oranges, and there would be a dedicated decoder for each apples and oranges, respectively.



report/assets/train.png

Figure 2: Learning the decoder D_X for domain X that uses embeddings from the shared latent space.

Style transfer. To perform style transfer, we simply combine the encoder E and a domain-specific decoder D_X or D_Y . For example, to obtain a mapping from domain Y to X , it suffices to use the encoder E and decoder D_X (Figure ??).

3.3 Training details

3.3.1 Layer dimensions

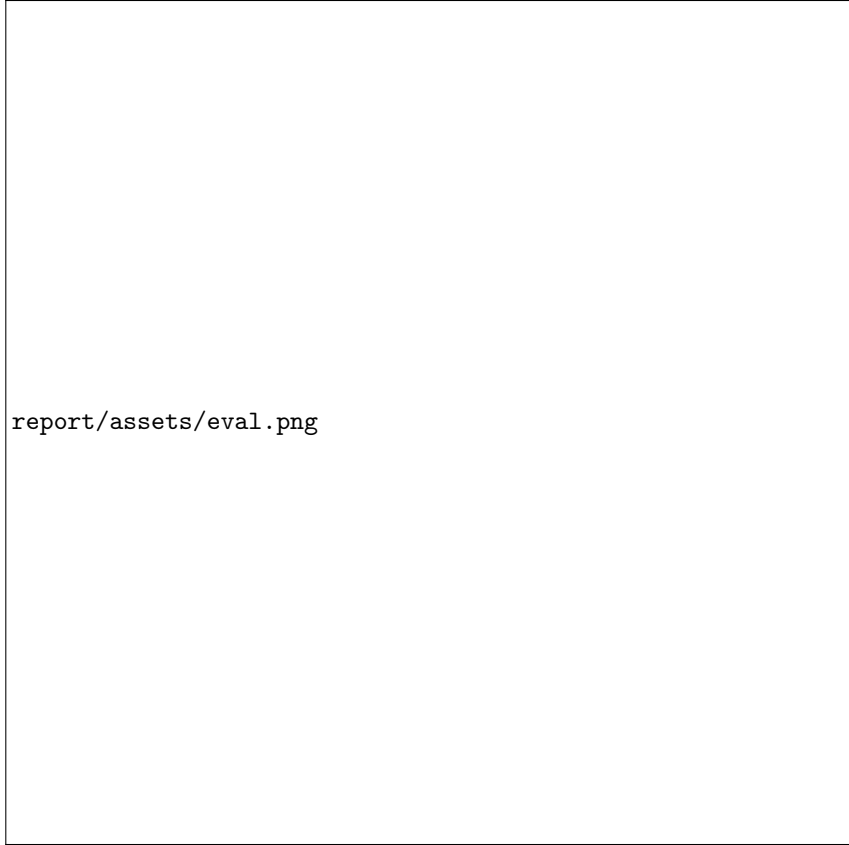
The shared encoder E and the domain-specific decoders D_x and D_y have the same size. Both the encoder and decoders are 5-layer deep with the following hidden sizes: 32, 64, 128, 256 and 512. The latent representation was a 256-dimensional vector. Each encoder and decoder layer included, in sequence, a convolution or transposed convolution, batch normalization, and the Leaky ReLU activation function. We use the Adam optimizer with a batch size of 64 and $\gamma = 0.95$. Hyperparameters used in the appendix.

3.3.2 Datasets

We trained the model on the apple2orange, summer2winter_yosemite and horse2zebra datasets. The images were split into domain-specific (i.e. apple and orange) datasets. The domain-specific datasets were used to train domain-specific decoders, and the combined domain dataset was used to train the shared encoder. The images used were of dimensions 256×256 .

3.3.3 Training

The model was trained on Google Colab using a single NVIDIA Tesla K10 GPU. The shared encoder and domain-specific decoders were trained for a maximum of 70 epochs, respectively. The training duration was approximately 80 minutes.



report/assets/eval.png

Figure 3: Performing style transfer from domain Y to X using the shared encoder E and domain-specific decoder D_X .

4 Experiments

We performed image translation experiments using three different datasets. The hyperparameters used to train on all the datasets can be found in Table ?? . The model results in translations that have some characteristics of the target domain, but the outputs still have blurry edges and lack clarity.

4.1 apple2orange

Our model’s performance on apple2orange dataset was promising as it learnt the shape and color of the fruits well. For instance, in the below translation of orange to apple, the translation removed the leaf and the stem of the orange. Qualitatively, the translations of orange to apple look better than vice versa.

4.2 summer2winter_yosemite

Our model’s performance on summer2winter_yosemite was not as good as apple2orange dataset. Qualitatively, we can tell the difference between the translations of the two seasons. The model learnt the color of the sky and the trees. The translations of summer images to winter resulted in output images that had gray skies and black trees, vice versa, the outputs had blue skies and green trees.

4.3 horse2zebra

Our model performed the worst on the horse2zebra dataset. Qualitatively, it is impossible to tell the difference between the translations of the two animals – the translations only show the silhouette.

It is possible that the model was not able to learn the physical features of the two animals since the horse dataset consists of horses of multiple colors.

5 Conclusion and future work

Our VAE architecture performs worse than CycleGAN or the related architecture proposed by ?]. While the output content (e.g. the rough horse and zebra silhouettes) is maintained, the output quality is subpar. Our domain-specific decoder was likely limited by the quality of the latent representation produced by the shared encoder.

A more complex encoder, such as the one proposed by ?], could learn a better representation of the domains and thus produce better translations. Another way to produce higher-quality images would be to replace our standard VAE architecture with a hierarchical VAE or VQ-VAE which has been shown to produce higher quality reconstructions. In addition, a more-informed loss function could improve quality. For instance, similar to ?], a loss term based on a discriminator network could be used for translated images.

A Appendix

A Hyperparameters

Table 1: The choice of hyperparameters

Hyperparameter	Value
Training batch size	64
Validation batch size	64
Learning rate	0.0005
γ (Adam)	0.95
α (\mathcal{L})	0.00025

B Individual contributions

The team collaborated closely across all phases of the project, including ideation, research, implementation, evaluation and writing. Adit experimented with hyperparameters and trained all the models that were used to generate the samples presented in the report. Haresh helped to implement the model. He contributed to the Abstract, Introduction, Training details, Experiments, and Conclusion sections of this report. Filip led the live coding sessions held by our group, created diagrams from sketches, referenced research in the Abstract, Introduction and Related works sections. He was also responsible for the bibliography.